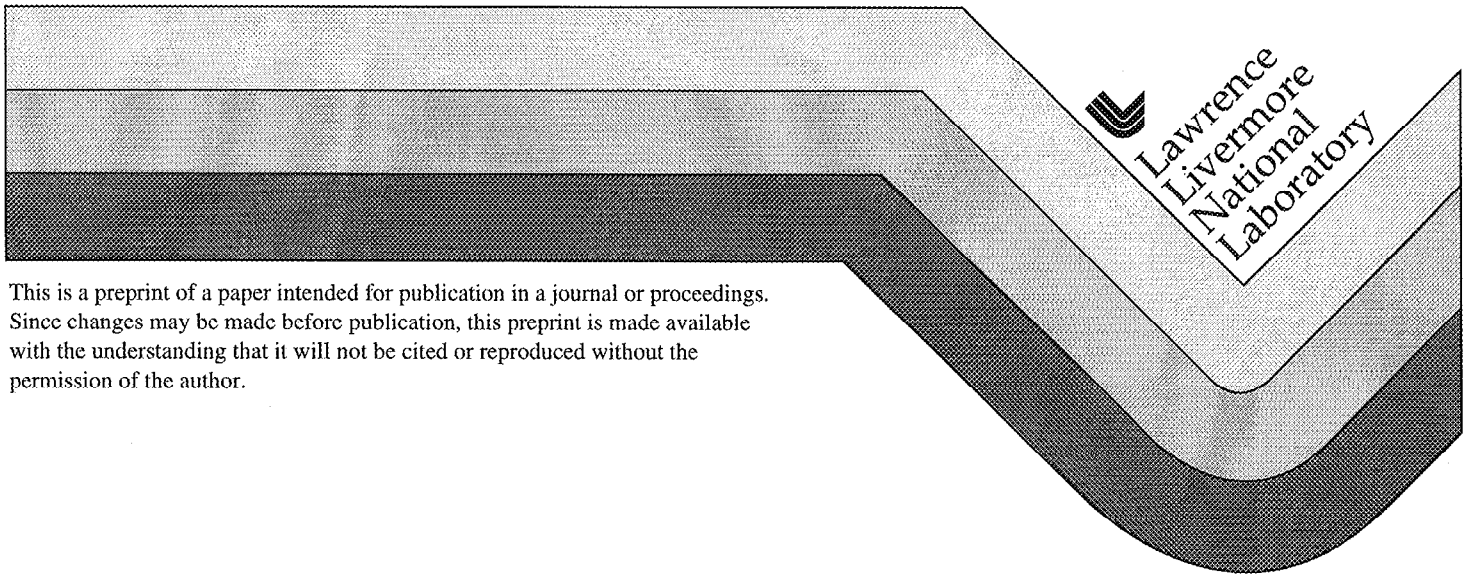
UCRL-JC-131349
PREPRINT

Petaflop Computing: Planning Ahead

James R. McGraw

This paper was prepared for submittal to the
Petaflop Operations Working Review (POWR) Workshop
Bodega Bay, CA
June 2-5, 1998

June 17, 1998



Petaflop Computing: Planning Ahead

James R. McGraw
Lawrence Livermore National Laboratory

This talk considers the problem of defining success criteria for petaflop computers. Current expectations for teraflop systems show an alarming acceleration of a trend we have seen for many years in high performance computers. Namely, it is becoming increasingly difficult to effectively use the computational capability of these machines. If this situation is not reversed quickly, the term "petaflop computer" may simply mean the next fastest computer that we cannot use. In many cases, we have some understanding of why we cannot achieve anywhere near the peak performance of these machines on real applications. Effective use of these resources is a highly complex optimization problem that must be solved over all of the different components of each application program. Given this complexity, it is the responsibility of our community to better quantify our progress in developing high performance systems with more meaningful metrics than simply "peak floating point operations per second." We need to develop metrics and tools that help us to enhance the end-to-end performance of solving large scientific applications on these advanced machines.

Based on current trends, teraflop systems of next few years could render the term "petaflop" practically meaningless. Going back to the days of vector supercomputing, it was common to find that large scientific applications could only achieve 30-70% of the peak rated speed of a machine. With distributed memory systems, the percent of peak often has been seen to be between 15-40%. For the new class of teraflop systems, some application developers have suggested that delivered performance on real scientific applications could be as low as 1-10%.

We have a general understanding of the causes of poor performance. As we scale up these systems, not all components are scaling at the same rate. Most architectures have included deep instruction pipelines, multi-level caches, greater numbers of processors, and more highly distributed partitions of main memory. At the same time, the computational algorithms have become less regular and less predictable in terms of their resource requirements. Techniques like adaptive meshes, unstructured grids, and Monte Carlo transport pose serious problems for dynamic load balancing and global communications. In the middle of these dramatic changes, key software tools like compilers and run-time systems have not been able to provide adequate solutions, so they continue to be a critical portion of the problem.

Given the "sins of the teraflops," the focus of improved metrics must center somehow on better conveying what level of performance a complete application can expect to achieve. The performance numbers described above depend precisely on problems being solved and the algorithms being used to solve them. Embarrassingly parallel algorithms and applications can often exceed these numbers while algorithms with non-trivial sequential sections will be much worse. The simple metric of FLOPS completely fails to convey

how well (or poorly) someone will be in using these machines to solve large scientific problems. If we are going to convince scientists and Congress to invest in petaflops, we will need to more clearly communicate what kind of true performance on real problems can be expected.

We can translate this long-term need into some specific areas of work that require immediate attention. These areas can help build up our understanding of the end-to-end performance of large applications on these machines. In this case, end-to-end performance refers to the process of evolving application demands into numerical algorithms that are translated into machine form and executed on a complex platform including processors, memory, networks, mass storage and I/O devices. One step to understanding end-to-end performance is to better characterize the potential performance of applications in terms of key system parameters. For example, what is the data locality of an algorithm as the size of the problem increases (possibly expressed as the working set miss rate as a function of the working set size). This kind of information could have a very beneficial impact on both cache hit rates and data partitioning across nodes. Likewise, another key step is in characterizing the performance of the full computer system in terms of relevant application parameters. For a given machine configuration, what level of cache hit rates, compute-to-communicate ratios, and compute-to-I/O ratios must be sustained to achieve different efficiency levels? This type of information flow between the applications and architecture developers will be necessary to enable even reasonable utilization of petaflop systems.

Petaflop Computing: Planning Ahead

James R. McGraw

Lawrence Livermore National Laboratory

Thesis for this Talk:

- Teraflop systems could render the term "Petaflop" meaningless.
- Effective utilization will reach an all-time low.
- $ 64,000 question: Where is all the potential going?
- Community must quantify progress with more meaningful metrics.
- Path to progress: end-to-end performance models.

Can We Use Systems Constructed from "Commodity-Components" to do Big Science ?

Question to ASCI Apps. :

For one of your real ASCI applications, attempting to use an entire Tflops machine, what percent of the "mythical" peak speed do you expect to achieve?

Answers from ASCI Apps. Developers

- To achieve above 10% use on ASCI Red Tflops, use assembly language for key loops. (Peak 25%)
- sPPM code (ASCI Blue ID benchmark), peak use is 16-18% for one processor. [with help from IBM]
- Full ASCI applications, planned for ASCI blue machines, the predictions for usage are very low.

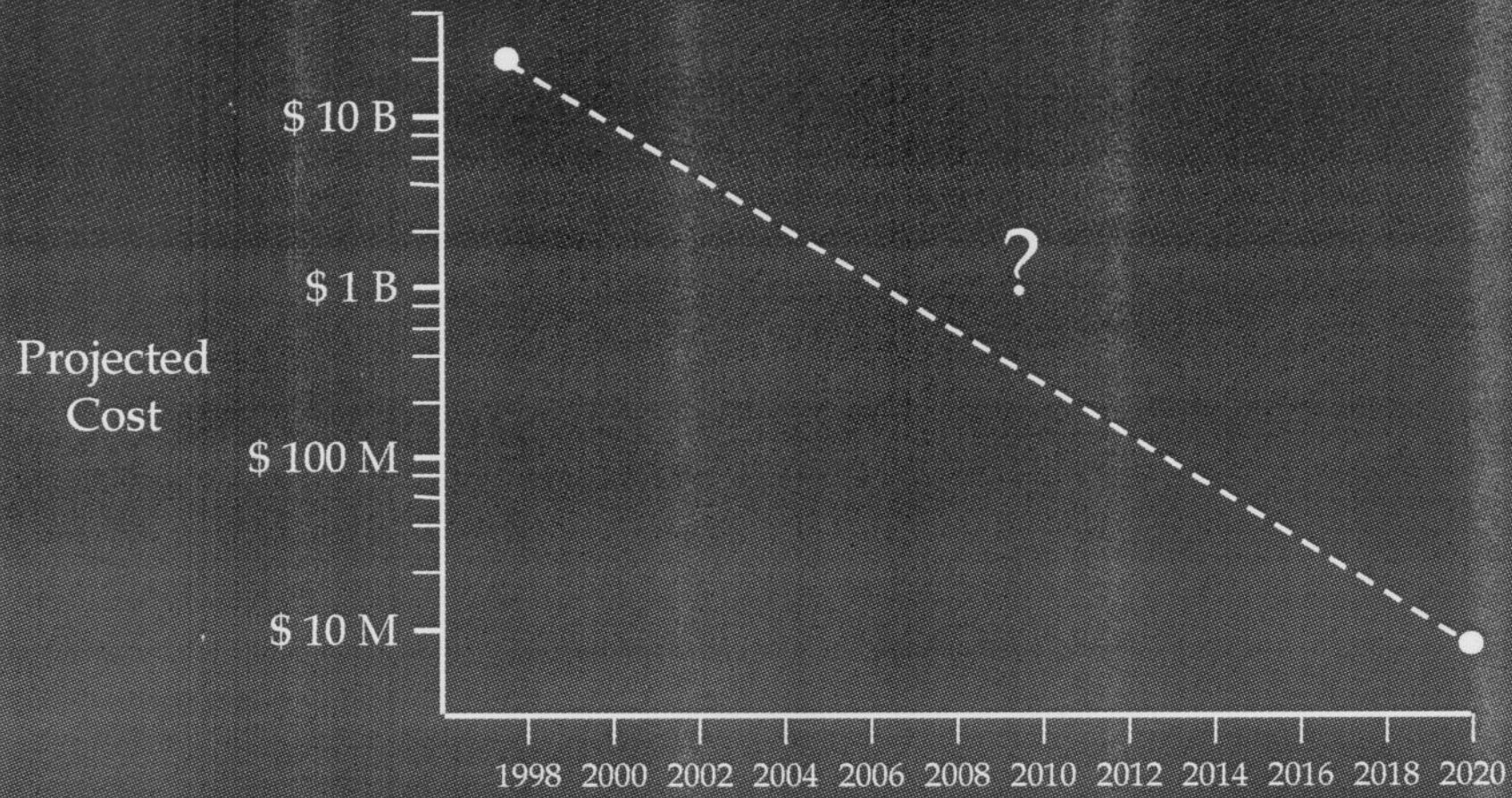
So How are We Losing All of the Capability?

- Divide instruction breaks instruction pipeline
- Local memory bandwidth inadequate for cache rates
- Poor "node" compiler performance
- Dynamic load imbalances
- Software message passing overhead
- Conservative algorithm design
- Non-scalable methods

Potential Corrective Measures

- Change computational algorithms / methods
- Reorganize data layouts / units of parallel work
- Extend compiler optimizations
- Expand tools for dynamic load-balancing
- Reduce off-chip communication & synchronization
- Change the architectural balance of commodity parts
- Enhance the performance of critical system parts

Petaflop Computing Is Doable: When? How Much? How Effective?



If Not "Petaflops" Then What?

- Ingredients for the right metrics are obvious:
  - » Total system cost
  - » Problem type and size that is solvable
  - » Total time to solution
- Defining them well is a tough challenge
- Selling them to the community, even harder

To Develop These Metrics Requires End-to-End Performance Insight

Necessary Areas of Contribution

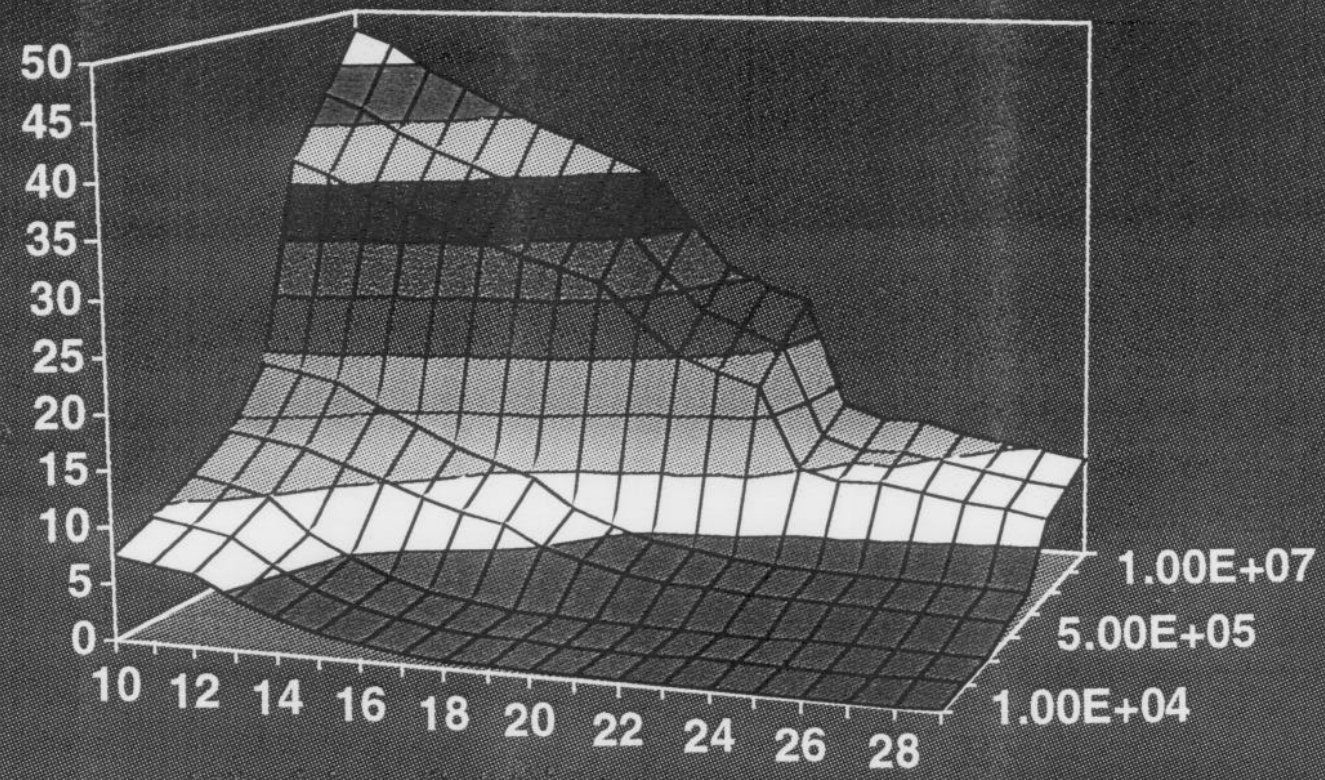
Apps.	Num. Alg.	System Software	Benchmark	Arch. / Networks	Mass Storage	Devices

Software Analysis

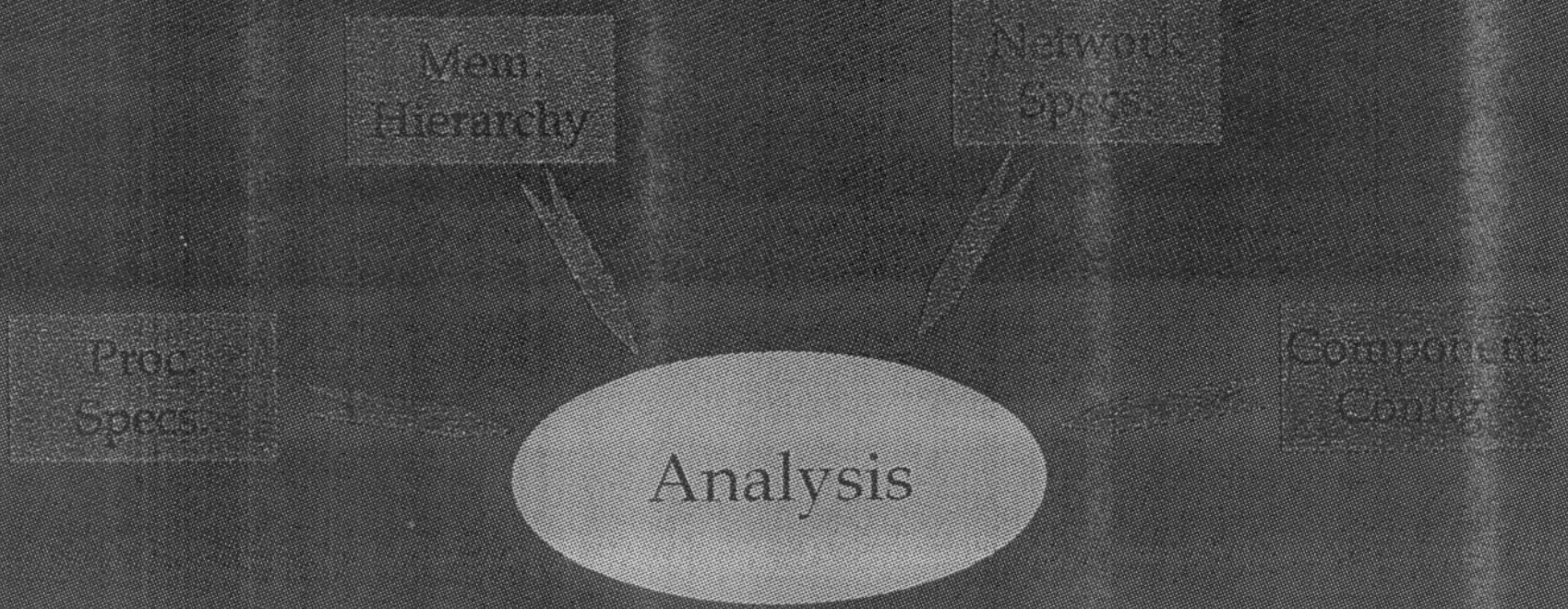
Performance Mgmt.

System Acquisition

Characterize Performance of Applications in Terms of Key System Parameters



Characterize Performance of Full "Systems" in Terms of Key Apps. Parameters

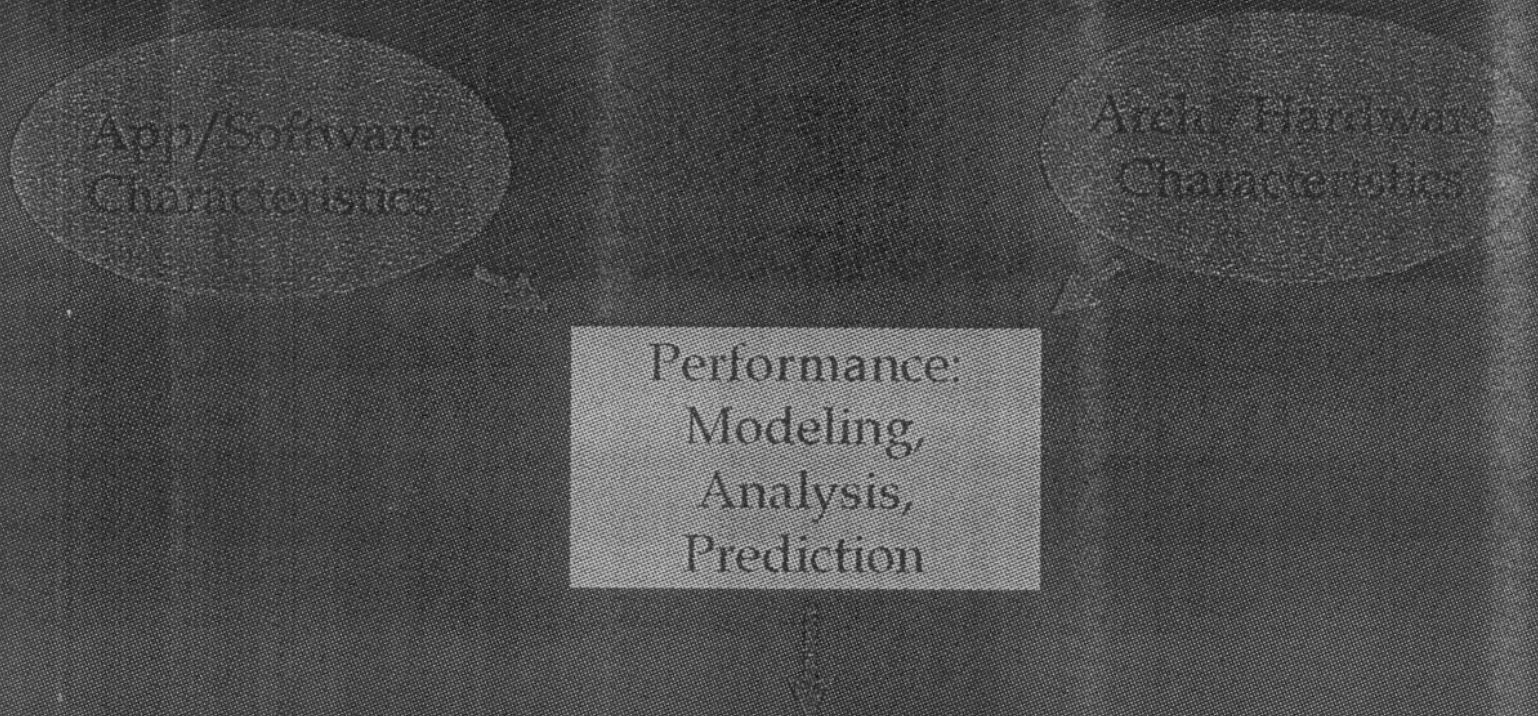


Compute-I/O rates

Cache hit rates

Compute-Communicate ratios

Building a Framework for Performance Management



Resource Requirements
Application Configurations
Performance Expectations

Conclusions

- The sins of the Teraflops will impact the push for Petaflops.
- Key to the future: more relevant and effective metrics for measuring performance.
- The path we must pursue is understanding the end-to-end performance of the systems we build coupled to the ways we use them.
- Yes, this is very hard, but for me, I like the hard challenges. That's why this field captures my interest.